Introduction to PL/SQL

- Jayendra Khatod

Objectives

- Introduction
- About PL/SQL
- PL/SQL Environment
- Benefits of PL/SQL
- Write PL/SQL Block
- Declare PL/SQL variables
- Execute a PL/SQL block

About PL/SQL

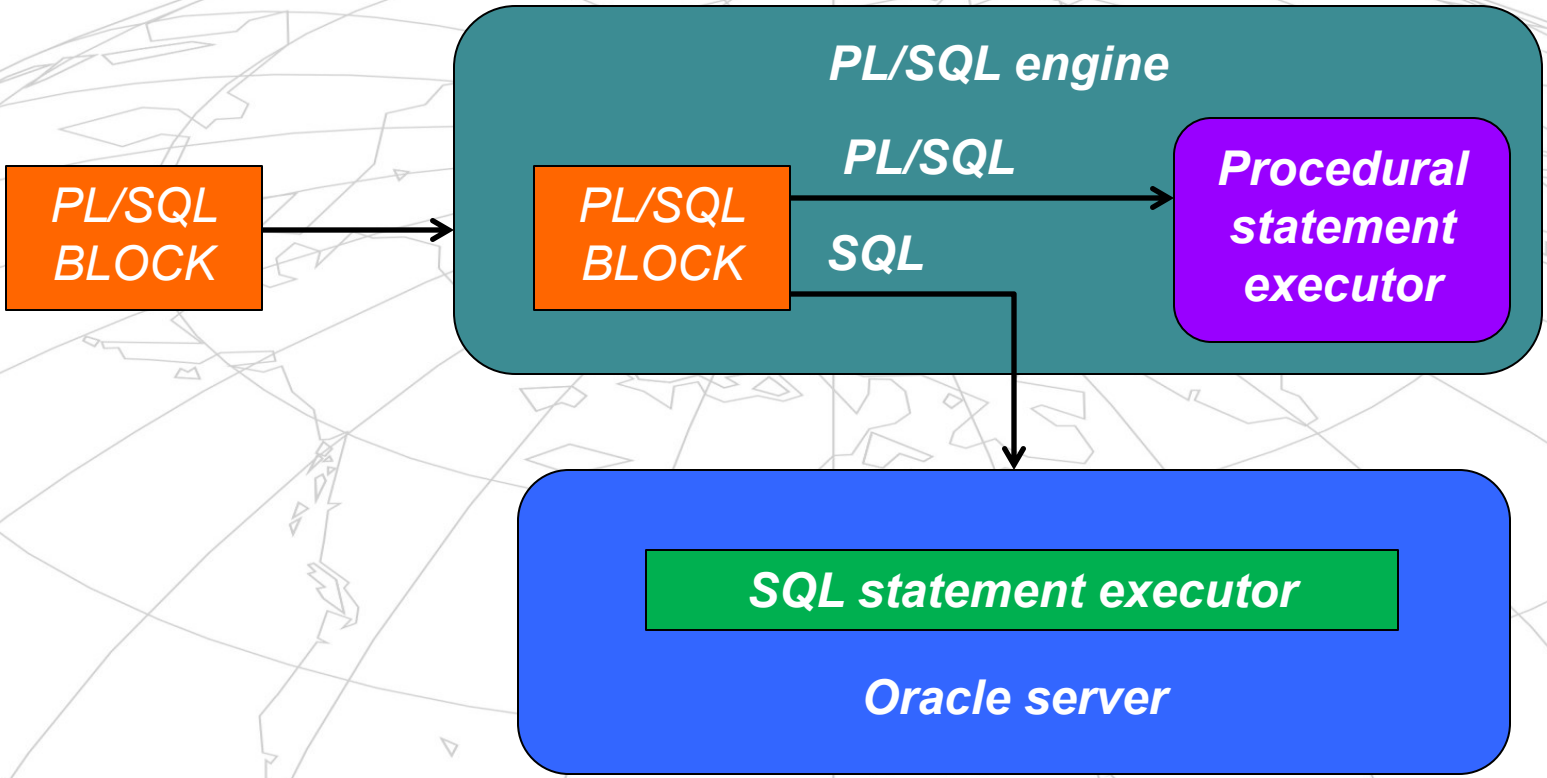
- **PL/SQL is Oracle Corporation's procedural language extension to SQL, the standard data access language for relational databases.**

- **PL/SQL offers modern software engineering features such as data encapsulation, exception handling, information hiding, and therefore brings state-of-the-art programming capability.**

About PL/SQL

- **PL/SQL incorporates many of the advanced features in programming languages**
- **With PL/SQL, you can use SQL statements to retrieve Oracle data and PL/SQL control statements to process the data.**

PL/SQL Environment



Benefits of PL/SQL

- You can reuse programs
- You can declare variables
- You can program with procedural language control structures
- PL/SQL can handle errors

PL/SQL Block Structure

```
DECLARE
   v_variable  VARCHAR2(5);

BEGIN
   SELECT  column_name
     INTO  v_variable
     FROM  table_name;

EXCEPTION
   WHEN exception_name THEN
   ...
END;
```

PL/SQL Block Types

Subprograms (Named Blocks)

Anonymous Block

```
[DECLARE]


BEGIN
   --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
   --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   --statements
   RETURN value;
[EXCEPTION]

END;
```

Use of Variables

- **Variables are used for:**
  - **Temporary storage of data**
  - **Manipulation of stored values**
  - **Reusability**
  - **Ease of maintenance**

Handling Variables in PL/SQL

- **Declare and initialize variables in the declaration section.**
- **Assign new values to variables in the executable section.**
- **Pass values into PL/SQL blocks through parameters.**
- **View results through output variables.**

Declaring PL/SQL Variables

Syntax

```
identifier [CONSTANT] datatype [NOT
NULL]
    [:= | DEFAULT expr];
```

Examples

```
Declare
  v_hiredate    DATE;
  v_deptno      NUMBER(2) NOT NULL := 10;
  v_location    VARCHAR2(13) := 'Atlanta';
  c_comm        CONSTANT NUMBER := 1400;
```

Declaring PL/SQL Variables

- **The Guidelines**
  - **Follow naming conventions.**
  - **Initialize variables designated as NOT NULL and CONSTANT.**
  - **Initialize identifiers by using the assignment operator (:=) or the DEFAULT reserved word.**
  - **Declare at most one identifier per line.**
  - **Two variables can have the same name, provided they are in different blocks.**

Assigning Values to Variables

Syntax

- *identifier := expr;*

Examples

Set a predefined hiredate for new employees.

```
v_hiredate := '01-JAN-04';
```

Set the employee name to Paul.

```
v_ename := 'Paul';
```

Variable Initialization and Keywords

- **Using:**
  - **Assignment operator (:=)**
  - **DEFAULT keyword**
  - **NOT NULL constraint**

Scalar Variable Declarations

- **Examples**

```
v_job          VARCHAR2(9) DEFAULT 'ADMIN';
v_count        BINARY_INTEGER := 0;
v_total_sal      NUMBER(9,2) := 0;
v_orderdate      DATE := SYSDATE + 7;
c_tax_rate     CONSTANT NUMBER(3,2) := 8.25;
v_valid        BOOLEAN NOT NULL := TRUE;
```

The %TYPE Attribute

- Declare a variable according to:
  - A database column definition
  - Another previously declared variable
- Prefix %TYPE with:
  - The database table and column
  - The previously declared variable name

Declaring Variables with the %TYPE Attribute

- **Examples**

```
...
  v_ename              emp.ename%TYPE;
  v_balance              NUMBER(7,2);
  v_min_balance          v_balance%TYPE := 10;
...
```

DBMS_OUTPUT.PUT_LINE

- **Example**

```
SET SERVEROUTPUT ON
DEFINE p_annual_sal = 60000
```

```
DECLARE
v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
v_sal := v_sal/12;
DBMS_OUTPUT.PUT_LINE ('The monthly salary is ' ||
TO_CHAR(v_sal));
END;
```

Commenting Code

- **Prefix single-line comments with two dashes (--).**
- **Place multi-line comments between the symbols /* and */.**
- **Example**

```
...
   v_sal NUMBER (9,2);
BEGIN
   /* Compute the annual salary based on the
      monthly salary input from the user */
   v_sal := &p_monthly_sal * 12;
END;      -- This is the end of the block
```

SQL Functions in PL/SQL

- **Available in procedural statements:**
  - **Single-row number**
  - **Single-row character**
  - **Datatype conversion**
  - **Date, General**

  } Same as in SQL

- **Not available in procedural statements:**
  - **DECODE**
  - **Group functions**

Summary

- **PL/SQL blocks are composed of the following sections:**
  - **Declarative (optional)**
  - **Executable (required)**
  - **Exception handling (optional)**

- **A PL/SQL block can be an anonymous block, a procedure, or a function.**
- **DBMS_OUTPUT.PUT_LINE**
- **SQL Functions in PL/SQL**

Thank You !